Turing machines

These notes describe Turing machines, a very simple model of a computer introduced by Alan Turing in his 1936 paper, "On Computable Numbers with an Application to the Entscheidungsproblem."

The basic model

The memory of the machine is a tape, divided into squares. Each square can hold one symbol from a finite alphabet of symbols. The blank is a special symbol, shown as `b` when we talk about it, but as the absence of a symbol in a square in pictures. The tape extends an arbitrary number of squares left and right. The tape alphabet is `b`, `0`, `1` in this part of the lecture. A tape with some symbols on it may be pictured:

```
-----------------------------------------------------
  |  0  |     |  1  |     |  0  |  0  |  1  |  0  |
-----------------------------------------------------
```

This is intended to show a portion of the tape containing the symbols `0b1b0010`. Symbols to the left and right of those shown are assumed to be blanks.

Access to the tape is via a read/write head, which is positioned at one of the squares of the tape. It may be pictured as a `^` symbol under the tape square where the read/write head is located, the scanned square. The symbol on the scanned square is the scanned symbol.

```
-----------------------------------------------------
  |  0  |     |  1  |     |  0  |  0  |  1  |  0  |
-----------------------------------------------------
                                  ^
```

In this diagram, we've added the `^` symbol to indicate that the read/write head is scanning the third `0` from the left. The machine can read and write the scanned symbol using its instructions (described below.)

In addition to the symbols stored on the tape, the machine may be in one of a finite number of states. Following Turing's usage, states are denoted by `q1`, `q2`, `q3` and so on.

In the picture, we'll put the symbol representing the current state underneath the symbol `^` representing the read/write head.

```
-----------------------------------------------------
  |  0  |     |  1  |     |  0  |  0  |  1  |  0  |
-----------------------------------------------------
                                  ^
                                  q3
```

Usually, the current state is shown in a little box, but that over-taxes our ASCII illustration skills. The contents of the tape, the location of the read/write head, and the current state of the machine together are a configuration.

In addition to tape, read/write head, and current state, the machine has instructions. Each instruction specifies what to do based on the current state and the symbol being scanned. "What to do" includes what the next state of the machine should be, what symbol to write on the tape in place of the symbol being scanned and whether to move the read/write head one square to the right or left.

An example of an instruction is:

```
If the current state is q3 and
the scanned symbol is 0, then
make q6 the next state of the machine,
write a 1 on the tape in place of the scanned symbol,
and move the read/write head one square to the left.
```

This instruction applies in the configuration given above, and the effect of executing it is as follows.

```
  ---------------------------------------------------
    |  0  |     |  1  |     |  0  |  1  |  1  |  0  |
  ---------------------------------------------------
                               ^
                               q6
```

We abbreviate instructions as 5-tuples. The instruction just considered is abbreviated as:

```
(q3, 0, q6, 1, L)
```

The order of elements in the 5-tuple are: the current state, the scanned symbol, the next state, the symbol to write, and the direction to move the read-write head (R for right, L for left.)

A Turing machine has a finite set of instructions. In any configuration, we require that there be at most one applicable instruction, that is, at most *one* instruction with a given current state and scanned symbol pair. That is, we require the machine to be deterministic. As long as there is an applicable instruction, it is executed to get a new configuration. If there is no applicable instruction, then the machine halts. The order in which instructions are listed in a Turing machine has no effect on their execution.

Running an example

Consider an example machine containing the instructions:

```
(q1, 0, q2, 1, R)
(q1, 1, q2, 0, R)
(q2, 0, q1, 0, R)
(q2, 1, q1, 1, R)
```

Suppose we run this machine starting with the following configuration.

```
 ----------------------------------------
   |     |  0  |  1  |  1  |  0  |     |
 ----------------------------------------
            ^
            q1
```

The applicable instruction is `(q1, 0, q2, 1, R)`, which specifies that that the machine go to state `q2`, replace the scanned symbol `0` by `1` and move the read/write head one symbol to the right. The resulting configuation is as follows.

```
-----------------------------------------
  |     |  1  |  1  |  1  |  0  |     |
-----------------------------------------
                ^
                q2
```

The applicable instruction now is `(q2, 1, q1, 1, R)`, which results in the following configuration.

```
-----------------------------------------
  |     |  1  |  1  |  1  |  0  |     |
-----------------------------------------
                      ^
                      q1
```

Now the applicable instruction is `(q1, 1, q2, 0, R)`, which results in the following configuration.

```
-----------------------------------------
  |     |  1  |  1  |  0  |  0  |     |
-----------------------------------------
                         ^
                         q2
```

Now the applicable instruction is `(q2, 0, q1, 0, R)`, which results in the following configuration.

```
-----------------------------------------
  |     |  1  |  1  |  0  |  0  |     |
-----------------------------------------
                            ^
                            q1
```

As there is no applicable instruction, the machine halts. Although we used each instruction at most once in this example, in general, each instruction may be used many times or not at all. In the homework, you will be asked to write a Turing machine simulator to carry out the task of running a given Turing machine starting with a given configuration. This particular machine flips every other symbol from `0` to `1` or vice versa, and leaves the symbols in between unchanged. For example, if the input is `111000`, the symbols on the tape when the machine halts are `010010`.

Suppose now we want to add states and instructions to this Turing machine to take the head back to the beginning of the string of `0` and `1` symbols. We add the states `q3` and `q4` and the following instructions.

```
(q1, b, q3, b, L)
(q2, b, q3, b, L)
(q3, 0, q3, 0, L)
(q3, 1, q3, 1, L)
(q3, b, q4, b, R)
```

Once the machine reaches a blank (`b`) in state `q1` or `q2`, it goes into state `q3` and moves left. In state `q3`, it moves left, leaving `0` and `1` unchanged until it reads a blank. Then it moves right into state `q4` and halts, because there is no applicable instruction.

A Turing machine to make a copy of its input

We now illustrate the process of designing a Turing machine to carry out a certain task. The task is to make a copy of the input string. We assume the machine will be started in state `q1` with its head on the leftmost

symbol in a string of `0`'s and `1`'s, and its task is to make a copy of that string and halt. In more detail, we specify that if the machine is started in the following configuation:

```
-------------------------------------------------------------
   |  1  |  1  |  0  |  1  |     |     |     |     |     |
-------------------------------------------------------------
      ^
      q1
```

the machine should halt with the following tape contents:

```
-------------------------------------------------------------
   |  1  |  1  |  0  |  1  |  c  |  1  |  1  |  0  |  1  |
-------------------------------------------------------------
      ^
```

Note that it has made a copy of its binary input string, separating the two copies by the symbol `c`, and returned the read/write head to the first nonblank symbol. We haven't specified what state the machine is in when it halts.

We'll choose to make the copy to the right of the input string (though that isn't part of the specification.) The first part of the computation will just run down the tape to the right until it finds the first blank, change it to the symbol `c`, and return the head to the leftmost nonblank symbol. That much of the computation is accomplished by states `q1` and `q2`, with the following instructions.

```
(q1, 0, q1, 0, R)
(q1, 1, q1, 1, R)
(q1, b, q2, c, L)
(q2, 0, q2, 0, L)
(q2, 1, q2, 1, L)
(q2, b, q3, b, R)
```

When the machine first reaches state `q3` in the example, the configuration will be as follows.

```
-------------------------------------------------------------
   |  1  |  1  |  0  |  1  |  c  |     |     |     |     |
-------------------------------------------------------------
      ^
      q3
```

Intuitively, the machine must then look at the current symbol, and move the read/write head to the right down the tape until the first blank and write a copy of the symbol it read. To keep track of whether it read a `0` or a `1`, the machine will use different states, namely `q4` or `q5`.

Another thing the machine has to keep track of is which of the input symbols it is currently copying. To do that, it will rewrite the symbol that it reads, replacing a `0` by a `d` and a `1` by a `e`. Later, it will rewrite `d` by `0` and `e` by `1` to restore the original input. So far, the instructions for state `q3` are as follows:

```
(q3, 0, q4, d, R)
(q3, 1, q5, e, R)
```

The job of state `q4` is to move down the tape, copying `0`'s, `1`'s and the `c`, until it gets to the first blank, where it will write a `0`. The job of state `q5` is similar to that of state `q4`, except that when it finds a blank it writes a `1`. With no further need to remember `0` or `1`, both states will go to state `q6`.

```
(q4, 0, q4, 0, R)
(q4, 1, q4, 1, R)
(q4, c, q4, c, R)
(q4, b, q6, 0, L)

(q5, 0, q5, 0, R)
(q5, 1, q5, 1, R)
(q5, c, q5, c, R)
(q5, b, q6, 1, L)
```

When the machine reaches state `q6` for the first time in the example, the configuration will be as follows.

```
-------------------------------------------------------------
   |  e  |  1  |  0  |  1  |  c  |  1  |     |     |     |
-------------------------------------------------------------
                            ^
                            q6
```

In state `q6`, the machine can move left copying `0`, `1`, or `c` until it reaches a `d` or `e`, which indicates the preceding copied symbol. It can change `d` back into `0` or `e` back into `1`, and then move right into state `q3`, to initiate the copying operation for the next symbol of the input.

```
(q6, 0, q6, 0, L)
(q6, 1, q6, 1, L)
(q6, c, q6, c, L)
(q6, d, q3, 0, R)
(q6, e, q3, 1, R)
```

The next time the machine reaches state `q3` in the example, the configuration will be as follows.

```
-------------------------------------------------------------
   |  1  |  1  |  0  |  1  |  c  |  1  |     |     |     |
-------------------------------------------------------------
         ^
         q3
```

Then the machine will change this `1` to an `e`, move right in state `q5` until the first blank, write a `1`, and enter state `q6` again. At that point, the configuration will be as follows.

```
-------------------------------------------------------------
   |  1  |  e  |  0  |  1  |  c  |  1  |  1  |     |     |
-------------------------------------------------------------
                                  ^
                                  q6
```

From here, the machine moves left until it reaches the `e`, which it changes back to a `1`, moving right into state `q3`. At this point, the configuration will be as follows.

```
-------------------------------------------------------------
   |  1  |  1  |  0  |  1  |  c  |  1  |  1  |     |     |
-------------------------------------------------------------
               ^
               q3
```

At this point, the machine changes the `0` to a `d` and moves right in state `q4` until it reaches the first blank, where it writes a `0` and moves left into state `q6`. At this point, the configuration of the machine is as follows.

```
-----------------------------------------------------------
   |  1  |  1  |  d  |  1  |  c  |  1  |  1  |  0  |     |
-----------------------------------------------------------
                                    ^
                                    q6
```

The next time the machine enters state `q3`, the configuration is as follows.

```
-----------------------------------------------------------
   |  1  |  1  |  0  |  1  |  c  |  1  |  1  |  0  |     |
-----------------------------------------------------------
                        ^
                        q3
```

The machine copies the `1` to the end of the string, and the next time the machine enters state `q3` the configuration will be as follows.

```
-----------------------------------------------------------
   |  1  |  1  |  0  |  1  |  c  |  1  |  1  |  0  |  1  |
-----------------------------------------------------------
                              ^
                              q3
```

At this point, the input is correctly copied, and there is no instruction specified for the machine in state `q3` with scanned symbol `c`. However, the problem specified that the head should be returned to the leftmost nonblank symbol, so we use a new state, `q7`, for that purpose:

```
(q3, c, q7, c, L)
(q7, 1, q7, 1, L)
(q7, 0, q7, 0, L)
(q7, b, q8, b, R)
```

These instructions will cause the machine to halt in the following configuration, as desired.

```
-----------------------------------------------------------
   |  1  |  1  |  0  |  1  |  c  |  1  |  1  |  0  |  1  |
-----------------------------------------------------------
      ^
      q8
```

This completes our construction of the Turing machine to copy its input. On the next page we give all of its instructions in one place, with comments.

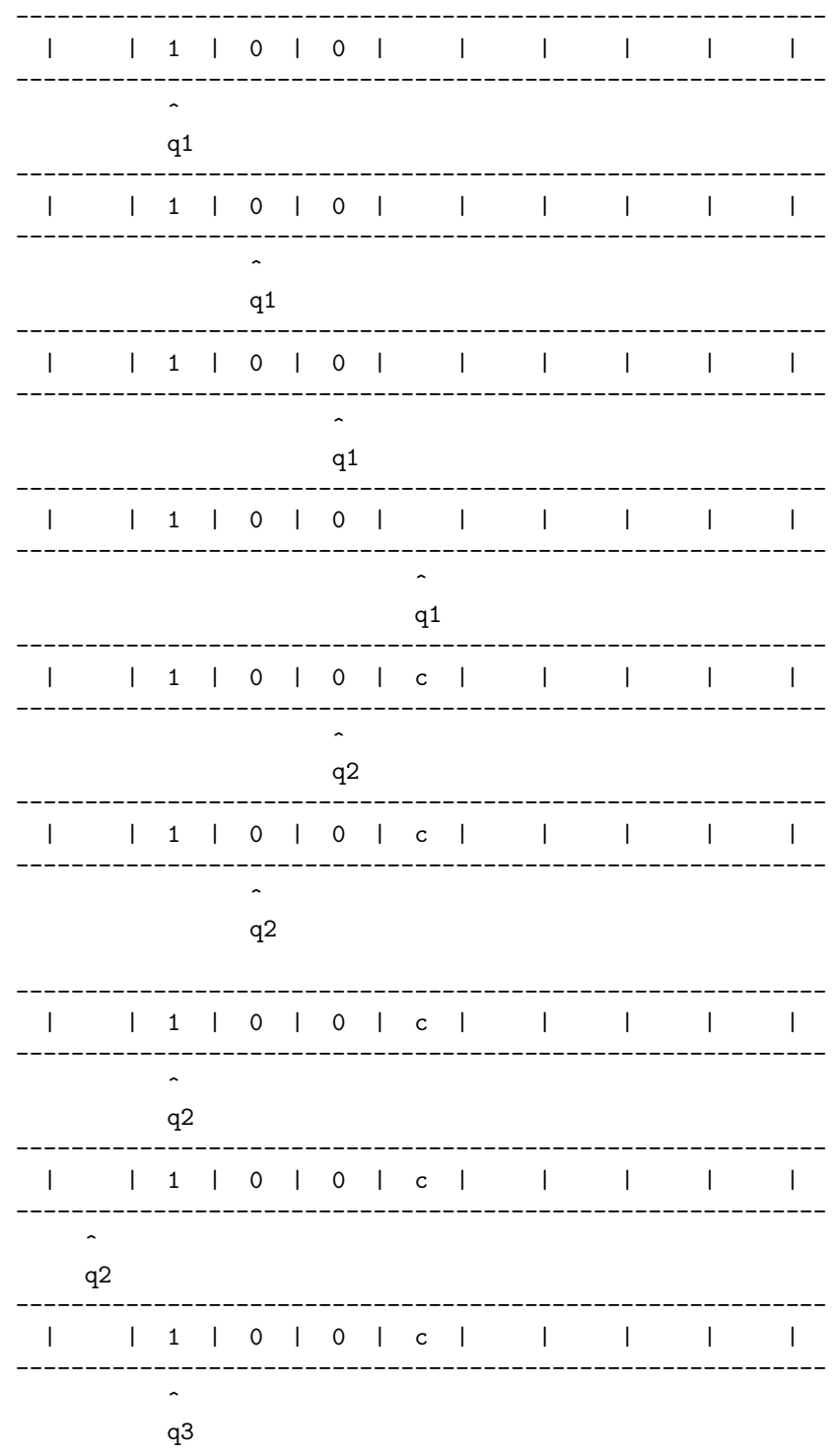
Turing machine to make a copy of its input

```
(q1, 0, q1, 0, R)           move right over 0's
(q1, 1, q1, 1, R)           move right over 1's
(q1, b, q2, c, L)           change first blank to c, move left to q2

(q2, 0, q2, 0, L)           move left over 0's
(q2, 1, q2, 1, L)           move left over 1's
(q2, b, q3, b, R)           at first blank, move right to q3

(q3, 0, q4, d, R)           change 0 to d, move right to q4
(q3, 1, q5, e, R)           change 1 to e, move right to q5
(q3, c, q7, c, L)           c means copying is over, move left to q7

(q4, 0, q4, 0, R)           move right over 0's
(q4, 1, q4, 1, R)           move right over 1's
(q4, c, q4, c, R)           move right over c
(q4, b, q6, 0, L)           replace first blank with 0, move left to q6

(q5, 0, q5, 0, R)           move right over 0's
(q5, 1, q5, 1, R)           move right over 1's
(q5, c, q5, c, R)           move right over c
(q5, b, q6, 1, L)           replace first blank with 1, move left to q6

(q6, 0, q6, 0, L)           move left over 0's
(q6, 1, q6, 1, L)           move left over 1's
(q6, c, q6, c, L)           move left over c
(q6, d, q3, 0, R)           change d back into 0, move right to q3
(q6, e, q3, 1, R)           change e back into 1, move right to q3

(q7, 0, q7, 0, L)           move left over 0's
(q7, 1, q7, 1, L)           move left over 1's
(q7, b, q8, b, R)           at first blank, move right and halt (q8)
```

On the next pages we illustrate a complete run of this machine.

A run of the copying Turing machine

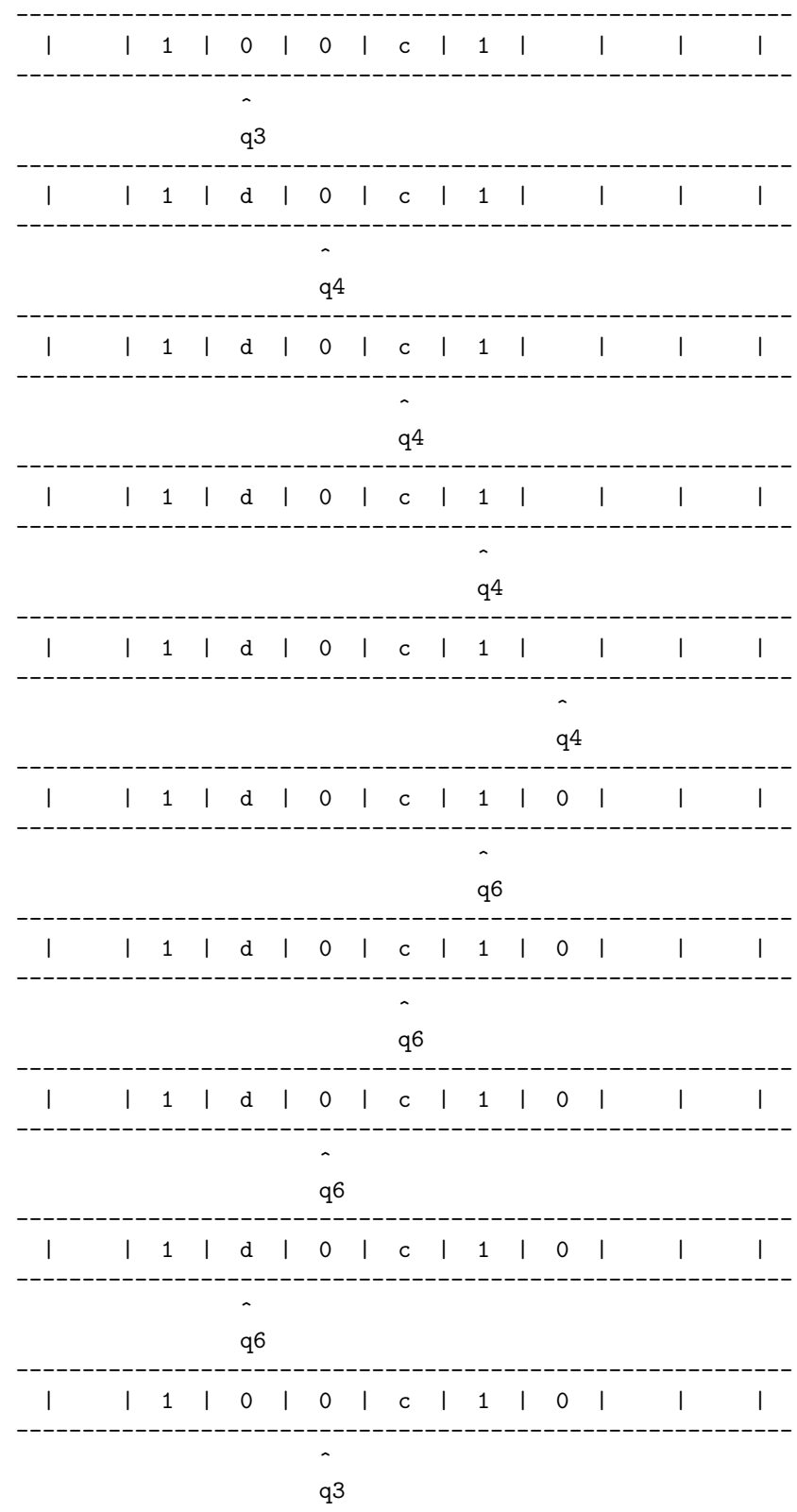
We exhibit a run of the copying machine on the input `100`. In the first phase, the machines adds a `c` to the end of the input and returns the head to the leftmost nonblank symbol.



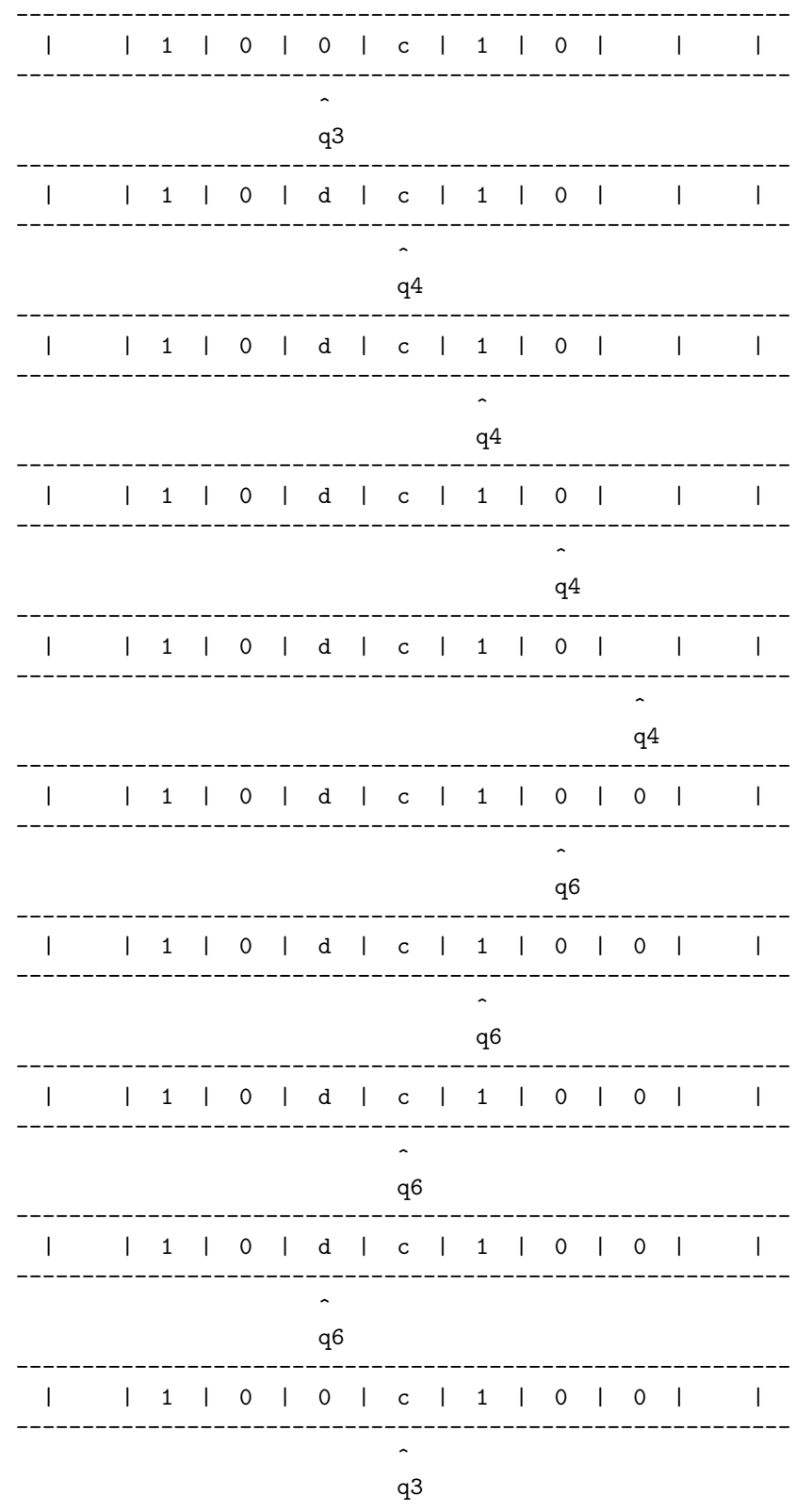
State q3 is the "main loop" of the machine, to note the next symbol, copy it to the end of the string, and get into position to copy the next symbol. (For convenience, we repeat the last configuration from the previous page.)

```
---------------------------------------------------------
  |     | 1 | 0 | 0 | c |     |     |     |     |     |
---------------------------------------------------------
          ^
          q3
---------------------------------------------------------
  |     | e | 0 | 0 | c |     |     |     |     |     |
---------------------------------------------------------
                ^
                q5
---------------------------------------------------------
  |     | e | 0 | 0 | c |     |     |     |     |     |
---------------------------------------------------------
                    ^
                    q5
---------------------------------------------------------
  |     | e | 0 | 0 | c |     |     |     |     |     |
---------------------------------------------------------
                        ^
                        q5
---------------------------------------------------------
  |     | e | 0 | 0 | c |     |     |     |     |     |
---------------------------------------------------------
                            ^
                            q5
---------------------------------------------------------
  |     | e | 0 | 0 | c | 1 |     |     |     |     |
---------------------------------------------------------
                        ^
                        q6
---------------------------------------------------------
  |     | e | 0 | 0 | c | 1 |     |     |     |     |
---------------------------------------------------------
                    ^
                    q6
---------------------------------------------------------
  |     | e | 0 | 0 | c | 1 |     |     |     |     |
---------------------------------------------------------
                ^
                q6
---------------------------------------------------------
  |     | e | 0 | 0 | c | 1 |     |     |     |     |
---------------------------------------------------------
          ^
          q6
---------------------------------------------------------
  |     | 1 | 0 | 0 | c | 1 |     |     |     |     |
---------------------------------------------------------
                ^
                q3
```

Next the machine copies the second symbol.



Next the machine copies the third symbol.



At this point, the machine moves left into state `q7`, which moves left copying symbols until the first blank, at which point it moves right and halts in state `q8` (which has no instructions.)

```
-------------------------------------------------------
  |     | 1 | 0 | 0 | c | 1 | 0 | 0 |     |
-------------------------------------------------------
                    ^
                    q7
-------------------------------------------------------
  |     | 1 | 0 | 0 | c | 1 | 0 | 0 |     |
-------------------------------------------------------
                ^
                q7
-------------------------------------------------------
  |     | 1 | 0 | 0 | c | 1 | 0 | 0 |     |
-------------------------------------------------------
          ^
          q7
-------------------------------------------------------
  |     | 1 | 0 | 0 | c | 1 | 0 | 0 |     |
-------------------------------------------------------
    ^
    q7
-------------------------------------------------------
  |     | 1 | 0 | 0 | c | 1 | 0 | 0 |     |
-------------------------------------------------------
          ^
          q8
```

Indeed the life of a Turing machine can get pretty tedious. This is a model of computation designed to be simple (and perhaps mathematically tractable) rather than programmer-friendly.